Lógica en la Informática / Logic in Computer Science

Tuesday April 22nd, 2014

Time: 1h55min. No books, lecture notes or formula sheets allowed.

1A) Let F and G be two propositional formulas such that $F \models G$. Is it true that $F \equiv F \land G$? Prove it using only the formal definitions of propositional logic.

Solution: It is true. The proof has two parts:
A) Let I be any model of F. We prove that then $I \models F \land G$. If $I \models F$ and $F \models G$, we have $I \models G$ (by definition of $F \models G$). Then $eval_I(F) = eval_I(G) = 1$. And then $I \models F \land G$ because $eval_I(F \land G) = min(eval_I(F), eval_I(G)) = min(1,1) = 1$.
B) Let I be any model of $F \land G$. We prove that then $I \models F$. $I \models F \land G$ implies $eval_I(F) = eval_I(G) = min(eval_I(F), eval_I(G))$, which implies that $eval_I(F) = eval_I(G) = 1$ and therefore $I \models F$.

1B) Given two propositional formulas F and G, is it true that either $F \models G$ or $F \models \neg G$? Prove it using only the formal definitions of propositional logic.

Solution: It is false. A counterexample is as follows: let F be the formula p and G be the formula q. Then $F \not\models G$: for example, if we define I s.t. $I(p) = 1$ and $I(q) = 0$ then we have $I \models F$ but $I \not\models G$. And $F \not\models \neg G$: now, if we define $I(p) = 1$ and $I(q) = 1$ then again $I \models F$ but $I \not\models \neg G$.

2) If S is a set of clauses, let us denote by $UP(S)$ the set of all literals that can be obtained from S by zero or more steps of unit propagation. Imagine you have a C++ program P that does unit propagation in linear time, taking as input any set of clauses S and returning $UP(S)$. Explain your answers to the following questions:
2A): Is it true that $l \in UP(S)$ implies $S \models l$?
Solution: Yes, if I is a model of given a clause $l \lor l_1 \lor \ldots \lor l_n$ and unit clauses $\neg l_1, \ldots \neg l_n$ then also $I \models l$, since $1 = eval_I(l \lor l_1 \lor \ldots \lor l_n) = max\{eval_I(l), eval_I(l_1), \ldots eval_I(l_n)\} = max\{eval_I(l), 0 \ldots 0\}$ which implies $eval_I(l) = 1$.

2B): Let l be any literal. Is it true that $S \models l$ implies $l \in UP(S)$?
Solution: No. Counterexample: if $S = \{p \lor q,\ \neg p \lor q\}$, then $S \models q$ but $q \notin UP(S)$.

2C): Can you use your program P to decide 2-SAT in polynomial time?
Solution: No. The program by itself cannot.

2D): Can you use your program P to decide Horn-SAT in polynomial time?
Solution: Yes, because a set of Horn clauses is satisfiable if and only if the output $UP(S)$ of P contains any pair of contradictory literals l and $\neg l$ (see also exercise 25 of "3. Deduccion en Logica Proposicional"):

If for some l, we have $UP(S) \supseteq \{l, \neg l\}$ then by 2A), we have $S \models l$ and $S \models \neg l$ and hence $S \models l \land \neg l$ so S is unsatisfiable.

For the reverse implication: if there is no l such that $UP(S) \supseteq \{l, \neg l\}$, then S is satisfiable, since it has the model I defined as $I(l) = 1$ iff l is a unit clause in $UP(S)$. This is true because Horn clauses have at most one positive literal, so there are only two possible kinds of clauses:

A) (one positive literal): for evey clause $l \lor C$ in S, if $I \not\models C$ then by unit propagation we have $l \in UP(S)$ and $I \models l \lor C$. and

B) (no positive literals): for every clause clause C of the form $\neg l_1 \lor \ldots \lor \neg l_n$ in S, if $I \not\models C$ then $I \models l_i$ for all i, so $l_i \in UP(S)$. But then by unit propagation also $\neg l_i$ would belong to $UP(S)$.

3A) Write all clauses needed to express the cardinality constraint $x_1 + \cdots + x_6 \leq 4$ without using any auxiliary variables (do not write any unnecessary clauses).

Solution: Of all subsets of 5 at least one is false:

$\neg x_1 \vee \neg x_2 \vee \neg x_3 \vee \neg x_4 \vee \neg x_5 \qquad \neg x_1 \vee \neg x_2 \vee \neg x_3 \vee \neg x_4 \vee \neg x_6 \qquad \neg x_1 \vee \neg x_2 \vee \neg x_3 \vee \neg x_5 \vee \neg x_6$

$\neg x_1 \vee \neg x_2 \vee \neg x_4 \vee \neg x_5 \vee \neg x_6 \qquad \neg x_1 \vee \neg x_3 \vee \neg x_4 \vee \neg x_5 \vee \neg x_6 \qquad \neg x_2 \vee \neg x_3 \vee \neg x_4 \vee \neg x_5 \vee \neg x_6$

3B) Write all clauses needed to express the Pseudo-Boolean constraint $1x + 3y + 4z + 5u + 8v \geq 14$ without using any auxiliary variables (do not write any unnecessary clauses). Hint: write one clause for each (minimal) subset S of the variables such that not *all* variables of S can be false.
Solution: $v, \quad u \vee z, \quad u \vee y, \quad z \vee y \vee x.$

4) We want to use a SAT solver to do *factoring*: given a natural number n, find two natural numbers p and q with $p \geq 2$ and $q \geq 2$, such that $n = p \cdot q$. Of course, the SAT solver will return "unsatisfiable" if and only if n is a prime number. (Curiosity: if we could factor large n, we could break many cryptographic systems!).

4A) Let a and b be bits (propositional variables). Write the seven clauses meeded to express that the two-bit number $c\,d$ is the result of the sum $a{+}b$, that is, c is the "carry" ($c = a \wedge b$) and d means "exactly one of a, b is 1" (exclusive or: $c = xor(a, b)$).

Solution:

$\neg c \vee a \qquad \neg c \vee b \qquad c \vee \neg a \vee \neg b$

$\neg d \vee \neg a \vee \neg b \qquad \neg d \vee a \vee b \qquad d \vee a \vee \neg b \qquad d \vee \neg a \vee b$

4B) Here we will factor numbers n of four bits $n_3\, n_2\, n_1\, n_0$ only, so $n \leq 15$. This means that, since we want to find $p \geq 2$ and $q \geq 2$, we know that $p < 8$ and $q < 8$ so for p and for q three bits each are sufficient, which we will call $p_2\, p_1\, p_0$ and $q_2\, q_1\, q_0$. Graphically, we can express the multiplication as we would do it "by hand":

$$\begin{array}{cccccc} & & & p_2 & p_1 & p_0 \\ & & & q_2 & q_1 & q_0 \\ \cline{4-6} & & & x_2 & x_1 & x_0 \\ & & y_2 & y_1 & y_0 & 0 \\ & z_2 & z_1 & z_0 & 0 & 0 \\ \hline & 0 & n_3 & n_2 & n_1 & n_0 \end{array}$$

using 9 intermediate auxiliary variables (called x, y, z, with subindices), where in fact we already know that z_2 must be 0. Using these auxiliary variables, and a few other auxiliary variables expressing the "carries" (please call them c_*), write here the expressions, like $n_1 = xor(x_1, y_0)$, cardinality constraints, etc., needed to ensure that indeed $n = p \cdot q$. After that, write the concrete clauses needed for each expression.

Solution: Since $x_0 = n_0$, we can directly define $n_0 = and(q_0, p_0)$. Every *and* of this kind generates three clauses as we wrote above for $c = a \wedge b$. We also have: $x_1 = and(q_0, p_1)$, $x_2 = and(q_0, p_2)$, $y_0 = and(q_1, p_0)$, $y_1 = and(q_1, p_1)$, $y_2 = and(q_1, p_2)$, $z_0 = and(q_2, p_0)$, $z_1 = and(q_2, p_1)$. Since z_2 must be 0, we need the clause $\neg q_2 \vee \neg p_2$.

We need two carry bits: $c_0 = and(x_1, y_0)$, $c_1 = atleasttwo(c_0, x_2, y_1, z_0)$, and also one clause $\neg c_0 \vee \neg x_2 \vee \neg y_1 \vee \neg z_0)$ (otherwise the sum is too large) and the bits for the result: $n_1 = xor(x_1, y_0)$ (four clauses as above), $n_2 = odd(c_0, x_2, y_1, z_0)$, and $n_3 = or(c_1, y_2, z_1)$. This last sum must give no carry: $atmostone(c_1, y_2, z_1)$. To encode this into CNF:

$c_1 = atleasttwo(c_0, x_2, y_1, z_0)$ can be expressed, e.g., making c_1 be the second output bit of a 4-bit sorting network, or with clauses: $\neg c_0 \vee \neg x_2 \vee c_1$, $\neg c_0 \vee \neg y_1 \vee c_1$, ... $\neg y_1 \vee \neg z_0 \vee c_1$, and $c_0 \vee x_2 \vee y_1 \vee \neg c_1$, $c_0 \vee x_2 \vee z_o \vee \neg c_1$, $c_0 \vee y_1 \vee z_o \vee \neg c_1$, $x_2 \vee y_1 \vee z_o \vee \neg c_1$.

$n_2 = odd(c_0, x_2, y_1, z_0)$ can be expressed by all 16 cases:
$\neg c_0 \vee \neg x_2 \vee \neg y_1 \vee \neg z_o \vee \neg n_2$, $\neg c_0 \vee \neg x_2 \vee \neg y_1 \vee z_o \vee n_2$, ... $c_0 \vee x_2 \vee y_1 \vee z_o \vee \neg n_2$.

$n_3 = or(c_1, y_2, z_1)$ can be expressed similarly to a binary or, with clauses:
$n_3 \vee \neg c_1 \qquad n_3 \vee \neg y_2 \qquad n_3 \vee \neg z_1 \qquad \neg n_3 \vee \neg c_1 \vee \neg y_2 \vee \neg z_1$.